Support Vector Machines and Generalisation in HEP

A. Bethani[1], A. J. Bevan[2], J. Hays[2] and T. J. Stevenson[2]

[1]LPSC, Grenoble, France. [2]Queen Mary University of London, London, United Kingdom.

E-mail: a.j.bevan@qmul.ac.uk, t.j.stevenson@qmul.ac.uk

Abstract. We review the concept of support vector machines (SVMs) and discuss examples of their use. One of the benefits of SVM algorithms, compared with neural networks and decision trees is that they can be less susceptible to over fitting than those other algorithms are to over training. This issue is related to the generalisation of a multivariate algorithm (MVA); a problem that has often been overlooked in particle physics. We discuss cross validation and how this can be used to improve the generalisation of a MVA in the context of High Energy Physics analyses. The examples presented use the Toolkit for Multivariate Analysis (TMVA) based on ROOT and describe our improvements to the SVM functionality and new tools introduced for cross validation within this framework.

1. Introduction

Machine learning methods are used widely within High Energy Physics (HEP). One promising approach, used extensively outside of HEP for applications such as handwriting recognition, is that of Support Vector Machines (SVMs), a supervised learning model used with associated learning algorithms for multivariate analysis (MVA). Developed originally in the 1960s, with the current standard version proposed in 1995 [1], SVMs aim to classify data points using a maximal margin hyperplane mapped from a linear classification problem to a possibly infinite dimensional hyperspace. However this means SVMs, like other MVA classifiers, have a number of free parameters which need to be tuned on a case by case basis. This motivates a number methods for ensuring the classifier is sufficiently generalised such that when used on an unseen dataset the performance can be accurately predicted. In this paper a brief overview of SVMs is given in Section 2, with an example using SVMs shown in Section 2.4. Generalisation is discussed in Section 3 with an illustrative example of how this can improve performance given in Section 3.3.

2. Support Vector Machines

2.1. Hard Margin SVM

Consider the problem of linear classification with the SVM where the training set, $\mathbf{x}$, is linearly separable. We define a separating hyperplane given by $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$, where $\mathbf{w}$, the weight vector, is perpendicular to the hyperplane, and b, the bias, determines the distance of the hyperplane from the origin (Fig. 1(a)). A decision function defined by $y_i = \text{sign}(\langle \mathbf{w} \cdot \mathbf{x_i} \rangle + b)$ is used to positively and negatively classify $\mathbf{x}_i$, the points in the training set. Without further constraint the choice of separating hyperplane is arbitrary and the geometric margin, γ, is introduced with the aim to maximise this. The functional margin, γ_i, for $\mathbf{x}_i$ is defined as

$$\gamma_i = y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1, \tag{1}$$

to impose the condition that no points lie within the margins. Considering two points that lie closest to the hyperplane referred to as support vectors (SVs), $\mathbf{x}^+$ and $\mathbf{x}^-$, with functional margins of $+1$ and -1 respectively, the geometric margin γ can be defined as $\gamma = \frac{1}{2}\left(\left\langle \frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot \mathbf{x}^+ \right\rangle - \left\langle \frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot \mathbf{x}^- \right\rangle\right) = \frac{1}{\|\mathbf{w}\|}$. The problem is solved in dual space by minimising the Lagrangian

$$\widetilde{L}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \left\langle \mathbf{x}_i \cdot \mathbf{x}_j \right\rangle , \tag{2}$$

for the parameters α_i (Lagrange multipliers) subject to $\alpha_i \geq 0$, where α_i is only non-zero for SVs, and $\sum_{i=1}^{n} \alpha_i y_i = 0$, providing the constraint equation. The full derivation leading to this minimisation and further details can be found in [2].

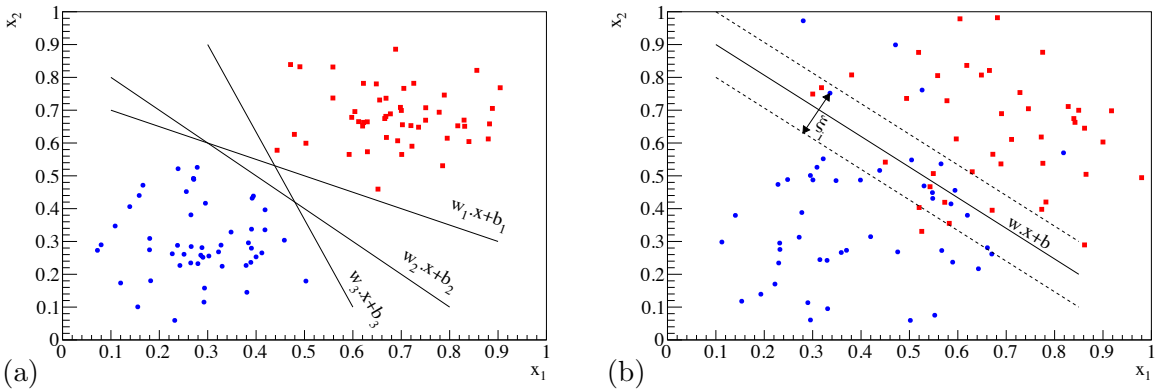

Figure 1: Illustrations of the (a) hard margin SVM where the solid line is the separating hyperplane, described by $\mathbf{w}$ and b, and the (b) soft margin SVM. The slack parameters, ξ_i, allow for misclassification of points weighted by the tunable cost parameter, C. Dotted lines are the margins, with the SVM finding the maximum geometric margin, γ.

2.2. Soft Margin SVM

The hard margin SVM is an important SVM algorithm, however it is not applicable in most real world cases. Issues arise if the data are not linearly separable due to being noisy, with data having small variations between different independent samples. This leads to the training becoming susceptible to over-fitting due to a few points in the training set, so some classification error must be allowed. To achieve this the hard margin constraint is relaxed and points are allowed on the incorrect side of the decision boundary, with the misclassification described by two additional parameters, ξ_i (slack) and C (cost), where ξ_i defines the distance from the correct margin to the ith incorrectly classified SV, and C is a tunable weight parameter multiplying the sum of slack variables penalising the misclassified points (Fig. 1(b)). This reduces to the same dual space Lagrangian problem as for the hard margin SVM, Eq. (2), however the constraint on the Lagrange multipliers is now $0 \leq \alpha_i \leq C$, while the constraint equation remains, $\sum_{i=1}^{n} \alpha_i y_i = 0$.

2.3. Kernel Functions and Feature Spaces

To this point the discussion of the SVM algorithm has been considered a linear decision boundary, however in most cases this is sub-optimal. In order to overcome this we can replace the inner product found in Eqns (1) and (2) with a kernel function (KF) $K(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}) \cdot \phi(\mathbf{y}) \rangle$ that maps the problem from the input space X to a potentially higher dimensionality, implicit feature space $F = \{\phi(\mathbf{x}) | \mathbf{x} \in X\}$, where the data may then be separable. In this process neither the feature map, $\phi(\mathbf{x})$, or the feature space, F, need to be known explicitly (known as the "Kernel Trick" [2]). This leads to specific properties required to define proper kernel functions in order for them to be used in this way. The kernel function should satisfy $K(\mathbf{x}, \mathbf{y}) = K(\mathbf{y}, \mathbf{x})$ and

$K(\mathbf{x},\mathbf{y})^2 \leq K(\mathbf{x},\mathbf{x})K(\mathbf{y},\mathbf{y})$. These are not strong enough conditions to guarantee the existence of F and so a KF is also required to meet Mercer's condition [3].

An example of a kernel meeting these requirements is the polynomial KF of Eq. (3) with two tunable parameters, the order d and offset c. Another common choice of KF is the Radial Basis Function (RBF), Eq. (4), where the distance between two SVs is computed and used as the input to a Gaussian with one tunable parameter $\Gamma = 1/2\sigma^2$. The RBF does not take into account potential bandwidth differences between dimensions in the input data which may cause tension and the norm between SVs may result in loss of information. It can be extended to include a tunable parameter, Γ_i for each dimension in the input space referred to as the multi-Gaussian KF, Eq. (5). The KFs implemented in TMVA (along with products and sums thereof) are:

$$K(\mathbf{x},\mathbf{y}) = (\mathbf{x}\cdot\mathbf{y}+c)^d = \left(\sum_{i=1}^{\dim(X)} \mathbf{x}_i\mathbf{y}_i + c\right)^d, \tag{3}$$

$$K(\mathbf{x},\mathbf{y}) = e^{-\Gamma\|\mathbf{x}-\mathbf{y}\|^2}, \tag{4}$$

$$K(\mathbf{x},\mathbf{y}) = \prod_{i=1}^{\dim(X)} e^{-\Gamma_i(\mathbf{x}_i-\mathbf{y}_i)^2}. \tag{5}$$

2.4. Higgs Boson Example: $H \to \tau^+\tau^-$

The Higgs Machine Learning Challenge dataset [4], with 6500 signal and 10000 background points is used to train SVMs with the KFs of Eqns (3-5), as well as a Boosted Decision Tree (BDT) [5]. The six variables offering the most discriminating power were used; MMC, $m_T(E_T^{miss}, l)$, $\Delta R(\tau, l)$, p_T^l/p_T^τ, ϕ Centrality and E_T^{miss}, see [4] for definition. The classifiers are optimised, trained and tested with TMVA [6] using the hold-out validation method, discussed in Sec. 3.1. Considering the receiver operating characteristic (ROC) curves, signal efficiency versus background rejection shown in Fig. 2(b), the classifiers all have similar performance for this problem. However it is not clear whether these solutions are fine tuned or optimal and so a further step is required to confirm if the results are generalised.

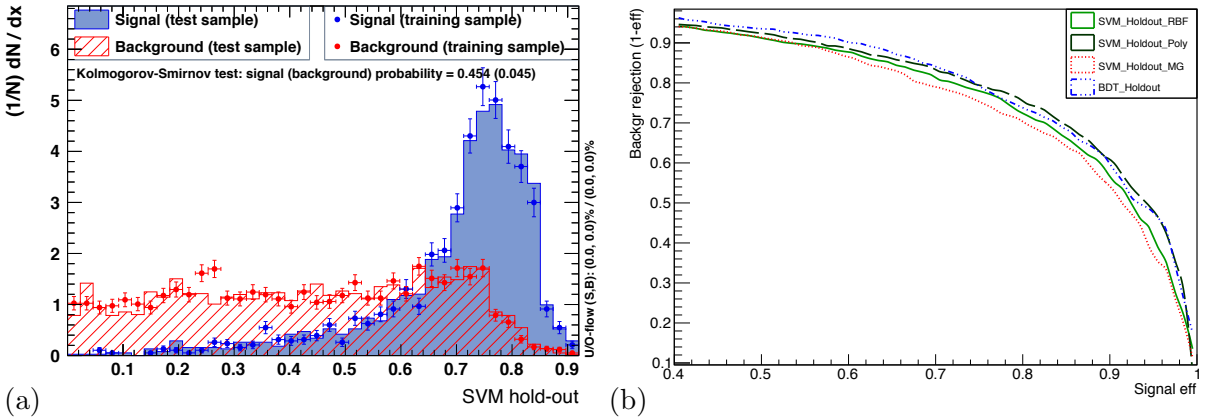

Figure 2: (a) TMVA outputs for SVM with RBF kernel optimised, trained and tested on the Higgs Machine Learning Challenge dataset with hold-out validation method. (b) ROC curves for SVMs with polynomial, RBF and multi-Gaussian kernel functions and a BDT.

3. Generalisation

Due to the often non-transparent nature of multivariate techniques one requires confidence that the trained MVA is robust against over-fitting, such that when used on an unseen data sample the performance of the MVA is accurately predicted and therefore the MVA is generalised. In

order to ensure this, validation techniques are required for both model selection, as most MVAs have at least one free hyper-parameter (HP), and for performance estimation, measured with a figure of merit such as the misclassification error. For an unlimited dataset these issues are trivial, as one could simply iterate through the HP space and models using a different subset of the data each time, evaluating the performance and choosing the model and configuration with the lowest error rate. In reality datasets are often much smaller than required. Naïvely the entire dataset could be used to select and train the classifier and to estimate the error, however this leads to over-fitting or over-training as the classifier learns specific fluctuations in the dataset and subsequently performs worse on unseen data. This is more distinct in classifiers with a large number of tunable HPs. This also gives an overly optimistic estimation of the error.

3.1. Hold-out Validation

A natural extension and potential way to overcome these issues is to split the dataset into two subsamples, one for training and one for testing. This is referred to as hold-out validation [7]. The training sample is used to select optimal HPs, using a method such as back propagation for the Multilayer Perceptron [8], and train the classifier. The testing sample is used to evaluate the performance. This method is not ideal as the performance differs depending on how the data is split, which may lead to misleading error estimates.

3.2. k-fold Cross-validation

In circumstances where a relatively small dataset is available it may not be possible to reserve a large portion of data for testing and so the hold-out method may not be viable. Instead the approach of k-fold cross-validation (CV) [7] can be used, performed as follows:

(i) Split the data into k equally sized, randomly sampled, independent datasets (or folds);
(ii) Train the classifier with the data from $k-1$ of the folds;
(iii) Test the classifier using the remaining fold;
(iv) Repeat steps (ii) and (iii) k times for all permutations.

The overall performance for the classifier is then given by the average error rate, $E = \frac{1}{k}\sum_{i=1}^{k} E_i$, where E_i is the misclassification rate for each training. This method has the advantage of using the entire dataset for testing and training, offering potentially improved performance over the previous methods.

However it raises the question of the number of folds to be used. A large number of folds will give good estimation of the average error rate, as the bias of the estimator is small, but the variance is large and the computational time is large, whilst the reverse is true for a small number of folds. In reality the choice of number of folds should be motivated by the size of the dataset available for the problem, for example a sparse dataset may require the extreme of leave-p-out CV [7] in order to train on as much data as possible.

Building on the k-fold method, ideally the data should be split into three statistically independent, randomly sampled datasets for training, testing and validation, with each set split into k-folds. The purpose of the training set is to choose the model and to optimise the HPs for this model through iterative k-fold CV, with one set of HPs for each fold. These HPs are then tested via k-fold CV with the validation set to obtain their estimated performance. The training and validation samples are then combined and used to train the final classifier with the best performing HPs. The test sample, which has been reserved until this point, is then used to assess the performance of this final fully trained classifier.

Taking the best performing classifier from the optimisation process may not be the optimal

solution or give the desired output, due to potential pathologies in the distributions. This also involves discarding a large number of trained classifiers from the process, so instead of using all HPs sets from the optimisation step, k classifiers can be trained and their performance estimated. These can then be used to give an averaged output on the testing sample which can help reduce fluctuations in the final distributions, improving agreement between the performance on the training and testing samples.

3.3. Higgs Boson Example: $H \to \tau^+\tau^-$

Using the Higgs Machine Learning Challenge dataset, discussed in Sec. 2.4, SVMs with RBF KFs were optimised, trained and tested using 5-fold CV to obtain the best performing and averaged SVMs, as well as using the hold-out validation method for comparison. The ROC curve comparing the different methods, Fig. 3(b), shows improvement for the k-fold CV classifiers over the hold-out method, both in the performance and shape of the curves. It is also worth noting that for the best performing and averaged classifiers from the k-fold CV, Fig. 3(a), show improved agreement between the training and test samples than for the hold-out validated method, Fig. 2(a), illustrating how this can lead to a more generalised result.

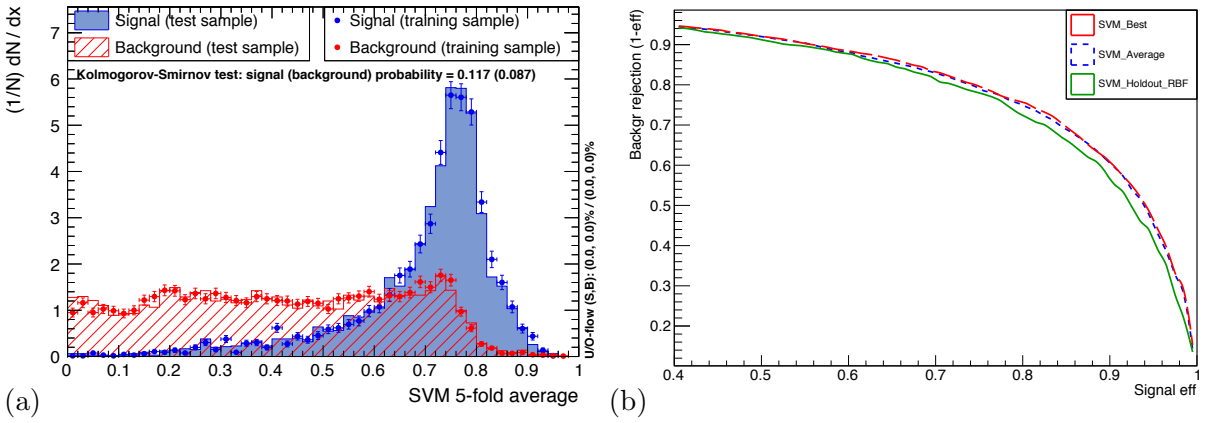

Figure 3: (a) TMVA output for a SVM with RBF kernel optimised, trained and tested on the Higgs Machine Learning Challenge dataset using 5-fold validation method averaging the classifiers from each fold. (b) ROC curves showing improved performance of the best and averaged 5-fold classifiers over the hold-out method classifier.

4. Summary

A brief overview of Support Vector Machines was presented, with an example showing similar performance to that of a BDT. However it is not clear without further checks as to whether the MVAs are sufficiently generalised. Hence a multistage cross-validation procedure has been outlined, which for the same example shows better performance as well as better agreement between the training and testing samples in the output distributions. This functionality is now available in the github version of ROOT.

References

[1] C. Cortes and V. Vapnik, Machine Learning Volume **20** Issue 3 (1995).

[2] J. Shawe-Taylor and N. Cristianini, Support Vector Machines and other kernel-based learning methods, Cambridge University Press (2000).

[3] J. Mercer, Philosophical Transactions of the Royal Society A **209** Issue 441−458 (1909).

[4] C. Adam-Bourdarios, G. Cowan, C. Germain-Renaud, I. Guyon, B. Kgl and D. Rousseau, "The Higgs Machine Learning Challenge," J. Phys. Conf. Ser. **664** (2015) No.7, 072015. See also URL: http://higgsml.lal.in2p3.fr/documentation/ (2014).

[5] L. Breiman, J. Friedman, R. Olshen, and C. Stone, Classification and Regression Trees, Wadsworth International Group (1984).

[6] A. Hoecker et al., PoS ACAT 040 (2007) 040.

[7] S. Arlot and A. Celisse, Statistics Surveys Vol. **4** 4079 (2010).

[8] R. Rojas, Neural Networks: A Systematic Introduction, Springer Science & Business Media (1996).